O.I.P.E. JC139
FEB 2 1 2002
PATENT & TRADEMARK OFFICE

03CO
#4

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	)	
	:	Examiner: Not Yet Assigned
ZHENYA ALEXNDER YOURLO	)	
	:	Group Art Unit: 2672
Application No.: 10/020,394	)	
	:	
Filed: December 18, 2001	)	
	:	
For: EFFICIENT VIDEO CODING	)	February 20, 2002

Commissioner for Patents
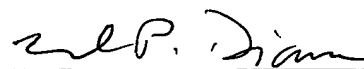Washington, D.C. 20231

CLAIM TO PRIORITY

Sir:

Applicant hereby claims priority under the International Convention and all rights to which he is entitled under 35 U.S.C. § 119 based upon the following Australian Provisional Application:

PR2126, filed December 18, 2000.

A certified copy of the priority document is enclosed.

Applicant's undersigned attorney may be reached in our New York office by telephone at (212) 218-2100. All correspondence should continue to be directed to our address given below.

Respectfully submitted,

_____
Attorney for Applicant

Registration No. _____

FITZPATRICK, CELLA, HARPER & SCINTO
30 Rockefeller Plaza
New York, New York 10112-3801
Facsimile: (212) 218-2200

#234581v1

10/020,394
2672

Patent Office
Canberra

I, GAYE TURNER, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PR 2126 for a patent by CANON KABUSHIKI KAISHA filed on 18 December 2000.

WITNESS my hand this
Twenty-fourth day of December 2001

GAYE TURNER
TEAM LEADER EXAMINATION
SUPPORT AND SALES

CERTIFIED COPY OF
PRIORITY DOCUMENT

ORIGINAL

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION FOR THE INVENTION ENTITLED:

Efficient Video Coding

Name and Address of Applicant:

Canon Kabushiki Kaisha, incorporated in Japan, of 30-2, Shimomaruko 3-chome, Ohta-ku, Tokyo, 146, Japan

Name of Inventor:

Zhenya Alexander Yourlo

This invention is best described in the following statement:

EFFICIENT VIDEO CODING

Field of the Invention

The present invention relates generally to video coding and, in particular, to an efficient approach to the provision of video information from an application source

5   operating from a computer network.

Background

Australian Patent Publication No. AU-A-53527/99 discloses a customisable user interface system, the salient components of which are illustrated in Fig. 1A. In this regard, Fig. 1A shows a hardware architecture of an interface system 100 where a smart

10   card 102 incorporating a memory arrangement is pre-programmed to facilitate user access to resources available via a computer network 105, such as the Internet. The smart card 102, is provided with a number of icons 104 or the like that are typically each representative of a particular function or access. The smart card 102 is insertable into a smart card reader 106 that is provided with electrical connectors 108 configured to couple

15   to complementary connectors (not seen in Fig. 1A) of the smart card 102 to enable a reading of the data in the memory arrangement thereof. The reader 106 is provided with a transparent touch panel 110 arranged so that when the smart card 102 is inserted into the reader 106 and electrical connection is made, each of the icons 104 are able to be viewed through the touch panel 110 whereby a user can depress the touch panel 110 at a location

20   overlying a particular icon 104 and the reader 106 operates to associate a position output from the panel 110 with a mapping stored within the memory arrangement of the smartcard 102. The reader 106 outputs a signal 112 associated with a function or some other predetermined event related to the selected icon 104. Typically, the reader 106 is a hand-held device and communicates with a computing arrangement, generally formed

25   within a so-called "set-top" box 114, that couples to a user output interface, in this example an audio-visual output device 116, such as a television set. The set-top box 114

operates to interpret the signals 112 received from the reader 106, which may be electrical, radio frequency, or infra-red, and according to a specific, possibly proprietary protocol. The set-top box 114 converts those signals to a form suitable for communication via the network 105 to cause appropriate transmission to a functional

5   destination, which may for example be a server computer 118. The server computer 118 performs the selected function, which in this case and according to the icons 104 of the particular card 102 illustrated, is the retrieval of on-line music, and provides data to the set-top box 114 which permits reproduction on the output device 116.

The system 100 is customisable by virtue of the user being able to utlize a

10   number of different smart cards 102 to perform corresponding different operations. For example, whereas the illustrated smart card 102 is used to retrieve and cause reproduction of on-line music video by way of the television set, other functions may be performed such as electronic banking, home shopping, ordering home delivery fast food such a pizza, and the like. In each instance, insertion of an appropriate smart card 102 into the

15   reader 106 causes a corresponding computer application to commence operation, either within the set-top box 114 or the server computer 118, in order to service user commands entered via the reader 106 and to return appropriate information for audio-visual feedback to the user. For example, associated with each of the above noted functions would typically be one or more menu displays which, in concert with the reader 106, form a

20   graphical user interface on the output device 116 by which the user can check selections being made (eg. pizza style to be ordered, toppings, payment methods) prior to actually confirming each or any function. An example of this is illustrated in Figs. 1B to 1D where, having inserted the smart card 102 into the reader 106, the application commences, for example on the server computer 118, and returns to the set-top box 114

25   for display on the output device a first menu screen 120 relating to the function to be performed, in this case a selection of Blues Guitar Masters. Using the reader interface

device 106 and by selecting appropriate icons 104, the user can scroll through the various offerings to make a desired selection, in this case for an artist called Young Dead Guy. A further menu screen 122 is then displayed as seen in Fig. 1C advising the user of the possible selections that may be made. The user again scrolls, and makes a desired

5   selection. The application then retrieves the selection, which in this case is a music video, which is then streamed to the set-top box 114 for appropriate output 124 as seen in Fig. 1D. Since the music video is, in effect, a series of "live" images, as compared to the substantially static images of the menu screens 120 and 122, the music video may advantageously be obtained and/or streamed from another location on the network 105

10   not associated with the generation of the menu screens 120 and 122.

It follows therefore that the set-top box 114, whether a simple "dumb" device, or an advanced device akin to a desktop or home computer, must be able to interpret and cause appropriate display and/or audio output on the device 116 of a range of data types, which may be sourced from direct video feed from the network 105 in the case of music-

15   video reproduction, static or animated computer graphical representations, bitmap images and perhaps via one or more various coding schemes. Further, different data types place differing demands on the network 105 and the application operating within the server computer 118 or the set-top box. Whilst in the described example, the ultimate product received is expensive in terms of network usage, it is always desirable to minimize the

20   amount of data required to be communicated over any network. The range of data types that may be used by various applications, dependent upon the smart card chosen by the user at any one time, complicates this issue in that some optimisation processes applicable to some data types may not be suitable for other data types. Related to this point is the general desired for communications to be realised in real-time. For video streaming, real-

25   time means the appropriate frame rate to ensure seamless viewing (eg. 25-30 frames per second depending on the format being used). However, for menu images and the like,

real-time may be any rate which presents the image at a user-acceptable rate, where there is little or no appreciable delay. In some instances this may, for example, be as long as the time taken for a user to look from a hand-held remote control device to the display unit, focus and absorb the information contained therein (say 0.1 - 2.0 seconds).

Summary of the Invention

It is an object of the present invention to substantially overcome, or at least ameliorate, one or more disadvantages of existing arrangements.

According to a first aspect of the present disclosure, there is provided a method of encoding a sequence of images for transmission over a communication network, said method comprising the steps of:

(i)   rendering a first image from a computer graphics application;

(ii)   encoding said first image according to a predetermined encoding scheme;

(iii)   rendering a second image from said computer graphics application;

(iv)   identifying changes between said first image and said second image from a change indicating output of said computer graphics application;

(v)   using said identified changes to determine a manner in which said second image is encoded according to said predetermined format; and

(vi)   repeating steps (iii) to (v) for each subsequent image of said sequence.

In accordance with another aspect of the present disclosure there is provided an interactive system in which user commands at a first location are communicated via a communications network to a processing arrangement at a second location, said processing arrangement interpreting said commands and in response thereto transmitting a sequence of images for display at said first location to said user, each said image being encoded to form at least a component of a data stream, characterised in that each said image is formed at said processing arrangement from a graphic object representation of

said image and a rendering of said graphic object representation is used to encode said image and form said stream.

In accordance with another aspect of the present disclosure there is provided an encoder for encoding a series of images into a bitstream, each said image being rendered

5   from a graphics object application characterised in that said encoder is constrained to operate according to a plurality of inputs at least one of which comprising a bitmap of each said image.

In accordance with another aspect of the present disclosure there is provided a system for manipulating user commands over a computer network, said system

10   comprising:

a user terminal coupled to said network at a first location and incorporating means for receiving commands from said user and display means for displaying images;

a server computer coupled to said network at a second location, said server computer comprising a computing application arranged to receive and process said

15   commands from said user terminal, and an output generator for forming an output display of said computing application;

characterised in that

said output generator comprises a graphic object application for rendering images formed by said computing application and an encoder for forming an encoded

20   bitstream output of said rendered images for communication over said network for display on said display means of said user terminal, and said user terminal comprises a decoder for decoding said bitstream for display on said display means.

According to another aspect of the invention there is provided a computer program product including a computer readable medium having recorded thereon a

25   computer program for implementing any one of the methods described above.

Other aspects of the invention are also disclosed.

Brief Description of the Drawings

One or more embodiments of the present invention will now be described with reference to the drawings, in which:

Fig. 1A is a schematic representation of a prior art user customisable interface system;

Figs. 1B to 1D show a number of examples of display output available from the system of Fig. 1A;

Fig. 2 schematically illustrates an operational architecture with which the system of Fig.1A may be operated;

Fig. 3 illustrates the updating of MPEG information;

Figs. 4A to 4E depict how various images may be updated;

Fig. 5 shows MPEG block information;

Fig. 6 shows how no change may be efficiently encoded;

Fig. 7 shows various conversions for different data structure;

Fig. 8 illustrates the predictive encoding of DC coefficients;

Fig. 9A shows the possible cases for neighbouring macroblocks;

Fig. 9B depicts Dirty and EncodeTouched flag marking;

Fig. 10 shows the marking of update dirty flags;

Fig. 11 shows a method of caching macroblocks;

Fig. 12 shows a macroblock updates using slices; and

Fig. 13 is a schematic block diagram of a general purpose computer upon which the arrangements described can be practiced.

Detailed Description including Best Mode

Traditional arrangements for delivering multimedia data over computer networks generally use some form of data compression or encoding to minimise transmission bandwidth. Such is particularly the case with Internet communications where static

images are often transmitted as JPEG images compressed using discrete cosine transforms or in other forms that use other transform methods. Audio information can also be compressed. There exists some forms of compression that accommodate both audio and visual information and this includes the various MPEG standards. As a consequence, the

5　arrangement of Figs. 1A to 1D may provide for at least the menus displays 120 and 122 to be delivered in a compressed form using any one of these known or similar standards. Typically, the video stream as seen in Fig. 1D may be delivered as pure video for example via a cable network, or alternatively in a compressed MPEG form, such a delivery method becoming popular over recent years. However, as indicated above, the

10　mere encoding of source information prior to transmission, whilst reducing communication bandwidth, does not necessarily optimise the volume of data to be communicated to a desired minimal level. Further, such encoding does not ensure, or even most substantially towards real-time display.

Fig. 2 shows an arrangement operable within the architecture of Fig. 1A which

15　provides for optimisations to be made to the data to be communicated via the network 105 in order to provide various outputs to the user using the output device 116.

As seen in Fig. 2, the set-top box 114 incorporates a card interface module 202 which interacts with the card reader 106 to interpret commands arising user operation of the card reader 106 and for forwarding those commands to an appropriate computer

20　application, such as one of the applications 210 coupled to the network 105. In some arrangements, it may be possible for one or more of the applications 210 to operate simultaneously and where this occurs a display manager 206 may be provided to manage the various outputs from the operating applications 210 so that an application 210 of precedence, for example corresponding to the smart card 102 currently inserted in the

25　card reader 106, has precedence for display on the output device 116. In this fashion, the display manager 206 operates with the knowledge of the various operating applications

210 to ensure that the application currently being used by the user has precedence on the output device 116. The display manager 206 communicates output information to the output device 116 via a output driver 204 formed within set-top box 114. The output driver 204 may include a display driver for driving a video or television display as well as an audio driver for driving an audio output device examples of both of which are found in traditional television receivers.

Fig. 2 also shows the salient features of a exemplary application 220 of the applications 210 which is seen to include three basic modules, being a process 222, an output generator 224 and an encoder 226. The process 222 couples to the network 105 via a connection 228 to receive commands from the set-top box 114 as well as source data from one or more other locations on the network as required by the particular process to be performed. The process 222 outputs to the output generator 224 data which is used to create the graphical user interface to be presented to the user via the output device 116. The information generated by the output generator 224 is then provided to the encoder 226 for compression prior to transmission via the network 105 to the output driver 204, typically via the display manager 206. With such an arrangement, a decoder, not illustrated but known in the art and substantially complementing the encoder 226, can operate to decode display data for reproduction using the output device 116.

It will be appreciated by those skilled in the art, that depending upon the particular nature of the application 220, the extent to which source data is derived from the network 105 and/or other communications occur via the network 105 as a result of an operation of the process 222 is completely application dependent. For example, if the application were related to the ordering of a home delivery pizza, commands originating from the card interface 202 would be used by the process 222 to identify the style of pizza being ordered, payment and delivery details, and the like. The process 222 may then output to the network 105 invoicing information which may be provided to a third party

such as a credit card facility (eg. operating as another of the applications 210) for debiting an account of the user ordering the pizza. Similarly, at the time the debiting is made, the actual order may be delivered to a local pizza manufacturing and delivery service. The provision of such data over the connection 228 relates to the specific process 222 and may

5   not always be bi-directional. For example, where the process 222 incorporates its own data storage relating to products to be delivered, for example the music videos of Fig. 1D, then there would be no requirement for bi-directional communication via connection 228 unless some form of on-line account debiting is desired.

As seen in Fig. 2, the output generator 224 provides graphics output data via a
10   connection 230 to the encoder 226 which provides an encoded output 232 for communication via the network 105, to the output driver 204 which incorporates a complementary decoder.

The preferred arrangement described herein makes use of the fact that many output generators 224 incorporate an intelligent image reproduction package such as a
15   graphical reproduction facility which has the capacity of handling graphical objects, incorporating including animations, text as well as bit map images. As seen in Fig. 2, the output generator 224 includes a graphics package 240 which, as is typical with such devices, outputs pixel-based information to a pixel mapping buffer 242. The pixel buffer 242 provides pixel-based data to the encoder 226 for conversion into the appropriate
20   encoded form, such as MPEG for example.

The preferred configuration also makes use of the number of facilities of the graphics package 240 so as to achieve optimised encoding, particularly using an encoding scheme such a MPEG, where the image is segmented and encoding performed on the segments. In this regard, the graphics package 240 is preferably configured to receive a
25   hierarchical tree-based display list 250 from the process 222 and from which the particular image to be generated is formed. The graphics package 240 interprets the tree-

based display list 250 to perform various compositing and/or rendering functions to provide the pixel-based data for storage in the buffer 242. In addition the graphics package 240 is configured to output further data indicative of changes made in the pixel buffer 242 for each separate screen or frame output. The change data is seen in Fig. 2 as an output 244 from the graphics package 240.

Also preferably, the graphics package 240 is configured to provide further change data relating to changes in the hierarchical tree 250 used to express the display list. Those changes are indicated as an output 246 of the graphics package 240. Each of the buffer change data 244 and the tree change data 246 are provided to the encoder 232 along with the pixel-based data from the buffer 242 and in the illustrated arrangement collectively form the connection 230. With this arrangement, the encoder 226 is able to perform optimised encoding in a manner to be described making use of the various attributes of the data and the changes being applied thereto.

In the MPEG image coding system, each macroblock of a coded stream has a "dirty" flag to indicate if and when the macroblock has been touched by an update. The dirty flag can be used to determine if re-encoding is necessary for an array of macroblocks that form a buffer of the previous frame. This is seen in Fig. 3 where an image 300 is divided into 16 (ie. 4x4) macroblocks and a dirty area 302 of 6 (ie. 3x2) macroblocks is indicated as requiring updating.

Below are described a number of approaches for improving the efficiency of MPEG encoding given a source in which the structure of the video to be encoded is known *a priori*.

In order for these optimisation to be understood it is appropriate to briefly review the manner in which MPEG encoding occurs. For each frame of video, a pixel map of image data in RGB format is initially supplied. The conversion of the pixel map to an MPEG data stream initially requires converting the RGB data into YCbCr data and

forming that data into macroblocks each of 16x16 pixels. Each macroblock is then considered to represent four blocks (each 8x8) pixels of luminance (Y) data, and a single downsampled block (each of 8x8 pixels) for each of chrominance blue (Cb) and chrominance red (Cr) data. A discrete cosine transform (DCT) is then performed on each

5 of these six blocks, giving a number of DCT transform coefficients including a DC coefficient and a number of higher frequency coefficients. The coefficients are then quantised and for the generation of I-frames the DC coefficients are differentially encoded, using the formula:

$$\Delta DC = DC - PDC$$

10 creating what are known as "DC deltas", along each block. A quantisation step is also performed. This is typically performed using run-level coding coupled with a table of variable length cells and results in an MPEG bitstream comprising the differentially encoded DC information and a limited amount of higher frequency information. MPEG data typically comprises Intra (I) frames, and two types of motion predictive frames

15 (P and B-frames). I-frames are used when a major change of information (screen content) occurs and typically transport the most data and therefore represent a (relatively) low compression rate. P-frames and B-frames are used for forward and backward prediction of information based on preceding I-frames, and may also be based on preceding P-frames, or for a B-frame, it may be based on a future P or I frame. The extent to which a

20 pixel frame is converted into an I-frame or a P-frame is generally a compromise between the extent of change from the previous frame and the complexity of the image, these factors influencing the decoding time at the set-top box, representing a trade-off between compression overhead and available bandwidth.

Initial I-frame Encoding

Applications in general have small number of "main" screens that may be displayed frequently, and to which small updates are made (for instance, changing a text label, or the state of a visible button).

The present optimisation provides a mechanism where a representation of the internal structures used to generate an I-Frame may be obtained by an application. The internal representation contains all the information necessary to rapidly construct an I-Frame representation of the screen with little to no computational overhead.

An application 210 may save and restore a semi-encoded I-Frame for each of its screens, thereby eliminating the need to encode a complete I-Frame from scratch each time the application needs to alter the look of the entire screen—also, once a semi-encoded I-Frame has been restored, subsequent updates are applied to the restored I-Frame representation, and hence only require re-encoding of the changed portions of the screen.

In a specific implementation, this optimisation involves encoding an initial I-frame which is cached with a corresponding tag. The I-frame is representative of a "background" upon which future encoded frames, such as the frames 120 and 122 are built. For example, as shown in Figs. 4A to 4E, an application tags four main screens (Figs. 4A-4D) that it uses. Each of these screens is encoded into an I-frame. Switching between the screens is allowed. Updates to each screen, such as the addition of text or other graphic objects can be encoded as one or more P-frames, or efficiently as individual I-frames.

In this fashion, as seen in Fig. 4E, an I-frame corresponding to Fig. 4A may be restored and updated with P-frames representative of text or other image components. Dependent on user selection, the background of Fig. 4C may then be restored and updated with P-frames as required. Use of the remaining backgrounds shown in Fig. 4 may follow in a similar fashion.

With reference to the example of Figs. 1B and 1C, each of the screens shown may be produced from a single cached I-frame corresponding to any background, and the foreground text and graphics (boxes), along with the highlighting of boxes selected by the user being derived from corresponding P-frames, one for each box option.

5   A further enhancement may be obtained as part of building applications that make use of the optimisation. In this regard, a method of automatically generating (offline) a pre-encoded set of screens (or a subset of the set of screens) that are used is desirable so that they may be restored at run-time with little computational overhead.

Further, a portion of an I-frame may be cached thereby permitting such a portion

10   to be later restored for use.

Super efficient I-frame encoding

In this optimisation, given that the changes made to a frame are known, it is possible to only encode those changes even if it is necessary to generate a complete I-frame. This may be achieved by caching an encoded version of each macroblock within a

15   macroblock structure so that if a macroblock is not "dirtied" by an update, the bit stream encoded version can be directly reused.

A representation of this is illustrated in Fig. 5 where an image 500 is divided into macroblocks 502 and for each macroblock 502 there is retained a record 504 including sufficient data enabling that macroblock to be replaced in a bitstream of macroblocks

20   where that macroblock may be dirtied by an update. In this fashion, rather than considering each frame as a single MPEG encoded bitstream, and caching the frame as in the above noted optimisation, the present optimisation provides for the frame to be cached on a macroblock by macroblock base whereby, as any one macroblock is dirtied, the updated encoded information for the new macroblock may be readily substituted into the

25   bitstream formed of the remaining macroblocks retained in the cache. As seen from Fig. 5, the relevant information necessary for retaining a cache of macroblocks includes

information representing the boundaries of the macroblock, "dirty" flag indicating whether the macroblock is an original macroblock or has been dirtied thereby indicating whether that macroblock should be used in the output bitstream, and the encoded representation of the macroblock incorporating encoded DC coefficients, encoded Dc

5   coefficients and a row of DC coefficients.

Efficient no-change bandwidth optimisation.

Typically, an MPEG-1 video stream requires coding of a constant number of frames per second. Generally, this would greatly increase the amount of computation that an encoder would need to perform within a given time period. However, it is possible to

10   take advantage of the characteristic behaviour of user interaction driven applications to alleviate this problem by way of encoding what is referred to herein as a "no-change" P-Frame.

A "no-change" P-Frame is encoded in cases where two successive frames to be encoded are identical, and may be encoded with very little computational overhead, and

15   also using only a small number of bits. Fortunately, applications 210 which rely on user-driven interaction, spend a large proportion of execution time waiting for user input, during which time the application 210 displays the same output. Accordingly, during this time, the required frame rate may be attained by generating a sequence "no-change" P-Frames until such time as the maximum limit for P-Frames is reached (at which point an

20   I-Frame must be coded), or until the application 210 needs to update the display.

A "no-change" P-Frame is coded by skipping all macroblocks in the frame that do not either form the start or end of a slice (by definition, macroblocks that form the start or end of a slice must be coded in a P or B-Frame. Each of the remaining macroblocks is coded using horizontal and vertical forward motion vectors with a size specified as zero

25   (indicating no motion), and with the macroblock address increment set to indicate the number of blocks that have been skipped between each coded block.

For a "no change" P-frame, a previously encoded first macroblock in the frame is used followed by the last previously encoded macroblock in the frame. Both macroblocks are cached in bit stream representation form. This converts the whole frame into one slice and is applicable to implementations in MPEG-1. For MPEG-2, each row of macroblocks becomes a slice.

An example of this is seen in Fig. 6 where a "no change frame" 600 is shown and which commences and ends with previously cached encoded macroblock 602 and 604 which, as illustrated are laid out or displayed in a rasterised fashion. As a consequence with this arrangement, since the set-top box 114 has received the previous frame, the set-top box 114 has a decoded version of that frame to be displayed. On receipt of information for the subsequent frame, the set-top box 114 decodes the P-frame noting that all macroblocks that are either the start or end of a slice have been skipped, and hence not changed, and also that the remaining macroblocks have been coded by forward motion vectors of magnitude zero, and hence have not changed. Such an approach provides for the display of each new frame where there has been no change using only a reduced bit-sequence. The specific advantage of this arrangement is that the decoding of the "same as previous" frame is performed within the set-top box and thereby provides for the application 210 to retain without any change the various macroblocks of the frame being displayed. In this fashion, where any macroblock is then updated, that macroblock can be readily changed necessitating the display of a new frame. Such ensures the continual transmission of the MPEG video bitstream without any need to intersperse control data that might otherwise be used to cause the set-top box to display again the "previous frame".

An extension of the above is shown in Fig. 12 where a frame 1200 is shown comprising macroblocks arranged in slices. Fig. 12 shows an untouched slice 1202. A further slice 1204 is shown which includes a dirty block 1206. In this case, since the dirty

block 1206 will make the sequence of DC deltas change in the current slice 1204, means that using the method of "Super Efficient I-frame Encoding" described above, to only quantise the DC can be employed for removing non-dirty macroblocks in the slice. In this regard, the macroblocks in the portion 1208 of the slice 1204 may be saved and

5   restored, whereas the portion 1210 must be reprocessed to determine the DC deltas thereof.

P-frame Encoding

The encoding of P-frames can operate on the same principal, according to the following method steps:

10   (a)   scan through the frame to find the first non-dirty macroblock;

(b)   push the cached version of the first non-dirty macroblock into the P-frame bitstream;

(c)   scan until the dirty macroblock is found;

(d)   encode the dirty macroblock;

15   (e)   repeat steps (c) and (d) as required until all dirty macroblocks have been encoded; and

(f)   push the cached version of the last macroblock into the P-frame bit stream.

Graphics Package

20   Further encoding efficiencies may be obtained through considering the manner in which the output generator 224 interprets commands from the process 222 to provide output information to the encoder 226. These types of efficiencies may be obtained in various forms and combinations from graphics packages dependent upon the facilities available in each package. For the purposes of the present disclosure, reference is made

25   to Australian Patent Publication No's. AU-A-47329/99 and AU-A-47339/99, the disclosure of each of which is hereby incorporated by cross-reference, which collectively

disclose a computer graphics rendering package which receives from a controlling process a display list for an image to be produced, the display list being formed as a hierarchical tree of graphical operations each working upon one or more graphical objects. The graphical objects may include individual graphical shapes, text,

5 combinations of shapes (ie. graphical sub-images), objects formed of pixel-based information, and the like. The graphics package is able to implement a number of graphical operations to achieve a compositing including transparency of the individual objects to provide for a wide range of desired outputs. Australian Patent Publication No's. AU-A-47329/99 and AU-A-47339/99 provides for a rendering package to be formed

10 having a number of types of output, such as:

(i) Pixel Map (pixmap)

(ii) Run Length Encoded (RLE); and

(iii) Plane Fill.

A pixmap requires use of standard MPEG encoding involving RGB to YCbCr

15 conversion for each pixel and then a Discrete Cosine Transform (DCT) of the converted image, as described above.

An RLE segment only requires RGB to YCbCr conversion to be performed once for each run, followed by a DCT. It is also possible to define a run in terms of a colour blend. For example:

20   Start colour: R_0, G_0, B_0

            Colour Delta: $\Delta R, \Delta G, \Delta B$

            n-th pixel in the run: $R_0 + n\Delta R, \quad G_0 + n\Delta G, \quad B_0 + n\Delta B.$

To achieve this blend in the YCbCr domain:

(i)   Convert $R_0 G_0 B_0$ to $Y_0 Cb_0 Cr_0$ including the required offset

25   of 128 for Cb_0 and Cr_0; and

(ii)   Convert ΔR, ΔG, ΔB to ΔY, ΔCb, ΔCr   without offsetting ΔCb or ΔCr by 128.

Therefore:

n-th pixel in the run: $Y_o + n\Delta Y, Cb_o + n\Delta Cb, Cr_o + n\Delta Cr$.

5   The conversion from RBG to YCbCr is a linear transform defined by:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1140 \\ -0.1687 & -0.3313 & 0.5000 \\ 0.5000 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

A plane fill does not require a Discrete Cosine Transform (DCT) since it is just a DC level, and requires only one RGB to YCbCr conversion. It will further be appreciated that the pixmap approach is the least efficient and that the plane fill approach is the most

10   efficient of the described alternatives.

This is seen in Figs. 7A to 7C. Fig. 7A shows a portion of a simple pixel map image in which there is no particular relationship between the colour displayed between adjacent pixels and the like. It follows that for such an image, individual conversion of each pixel is required. In Fig. 7B, it is apparent that the pixels are arranged in runs of

15   various lengths and that for each one, only a single conversion need be performed. Accordingly, for Fig. 7A whereas a total of 12 conversions would be required, Fig. 7B would require six conversions, these being the total number of runs for the display lines indicated. In Fig. 7C, the plain fill has only a single pixel value for the entire graphical object. As a consequence, only a single conversion need be performed and the

20   conversion need only be limited to obtaining the DC value since there is no higher frequency content across the entire macroblock.

Motion Prediction

This optimisation makes use of the original tree 250 from which the image is composited, and the transformation matrices that correspond to the nodes in the tree 250 ,

it is also possible to predict the motion of individual components of a scene to enable greater compression.

For example, returning to Fig. 2, it is seen that the nodes in the expression tree 250 are particular graphical operations which act upon the graphical elements (A, B, C, D, and E) to achieve the desired composited output. The arrangements disclosed in Australian Patent Publication No's. AU-A-47329/99 and AU-A-47339/99 provide information of changes to the expression tree 250 which are output 246 as illustrated. Those changes may include, for example, the omission of the graphical object B from the tree 250, resulting in a simplified expression tree incorporating the operation *A in C* as one component thereof. With such changes in the expression tree 250 being flagged at the output 246, the encoder 226 is able to identify those macroblocks in which the graphical object B was represented and therefore is able to immediately dirty those macroblock and provide only for the re encoding of those macroblocks as output from the graphics package 240 via the pixel map buffer 242. Accordingly, even where there exists a change in the expression tree 250 which creates a new frame of output, again only those portions of the output that change need be identified. Specifically and importantly, the use of the graphics package to identify those portions that have changed as opposed to some other detection arrangement greatly increases the speed at which encoding can be performed and therefore the rate at which data can be delivered to the set-top box 114.

Handling Neighbouring Macroblocks

For each macroblock, the DC coefficients are encoded predictively as illustrated in Fig. 8. As seen in Fig. 8, encoding of a I-frame occurs on a macroblock-by-macroblock basis whereby each of the YCbCr components are processed according to their division into the macroblock. As discussed earlier, for each macroblock there are four luminance (Y) components and the two separate chrominance (Cb, Cr) components. As seen from Fig. 8, each of the luminance components for a macroblock are

progressively encoded into the I-frame and then encoded in sequence onto the luminance components for the next macroblock. A similar arrangement occurs with each of the chrominance components.

Accordingly, for the cases in which some (neighbouring) blocks are dirtied, and require re-encoding, it is necessary to treat the DC coefficients in a special manner to ensure that they are correctly encoded. Fig. 9A shows the four possible cases for dirty and non-dirty neighbouring macroblocks. The present inventor has therefore determined that within each macroblock, it is desirable to have a pointer to the previous block. Further, it is also desirable to store the current DC value within each block so that it can be obtained as needed. For some cases, it is possible to avoid quantisation of macroblocks and reuse the bit stream corresponding to an individual macroblock.

When encoding I-Frames, the DC component of each macroblock is referenced from the previous macroblock in scanline order:

$$\Delta DC = DC - PDC$$

If the macroblock is the first macroblock in a slice, the DC value is reset to a known initial value (eg. 1024 in MPEG-1). This means, that if there is an uninterrupted sequence of non-dirty macroblocks from the start of a slice, then it is possible to simply restore a previously cached bitstream version of each macroblock that comprises the sequence. If, however, a macroblock after the first macroblock in a slice is marked as dirty (and hence needs re-encoding), the cached bitstream representations of subsequent macroblocks cannot be used, since the chain of DC delta values will no longer be correct.

This situation may be corrected by using a cached, quantised version of each macroblock subsequent to one that has been encoded and to only update and quantise the DC component. Each of these macroblocks may then be coded into the bitstream, restoring the chain of DC delta values.

In order to facilitate this optimisation, and as illustrated in Fig. 9B, an EncodeTouched flag is used. The EncodeTouched flag is set on any macroblock that has been previously marked as dirty (and subsequently re-encoded), and also on any block for which the previous block has had its EncodeTouched flag set. This allows the encode function to check whether the previous macroblock, in scanline order, has had its DC value modified, so that a decision can be made as to whether to use the cached bitstream representation, or cached quantised versions of macroblocks comprising the remainder of the slice.

Update Dirty Flag Numbering

It is necessary to determine the set of macroblocks touched by an update. This is seen in Fig. 10 where a screen display 1000 has macroblocks 1002 and an update region 1004 as indicated. Within the region 1004 it is appropriate that each of the macroblocks that have been dirtied and which to define the region 1004 should be numbered in an orderly fashion so that the corresponding macroblocks of the previous frame can be readily identified and substituted with newly encoded macroblock information.

Such necessitates an efficient method for caching macroblocks that are to be used as part of an I-frame, as discussed below.

Caching Methods

A method for caching macroblocks for reuse initially requires storing the delta for the DC component of each part of the macroblock (4 x YCbCr) within the macroblock structure. Also stored is a reference from each macroblock to the previous macroblock, as discussed above. This is illustrated in Fig. 11 where each macroblock structure includes a reference pointer to previous macroblocks in the stream, forming a chain of reference pointers between macroblocks in reverse scan order.

After each macroblock has been transformed using a DCT, the macroblock is quantised and stored for later retrieval. Each macroblock is also given a "dirty" flag, which may be marked is either "dirty" or "not dirty".

For each macroblock that is marked as dirty, the following steps are performed:

5
(i)   apply a DCT;

(ii)   calculate the DC delta ($\Delta DC = DC - P$) and save the DC values;

(iii) quantise the macroblock;

(iv)   save the macroblock;

(v)   encode the macroblock (bit stream encoding);

10
(vi)   save the bitstream version; and

(vii) set dirty flag to false.(ie: not dirty).

For each macroblock that is marked as not dirty, and is not preceded by dirty blocks since the start of the current slice, the macroblock can be restored from the bitstream encoded version.

15
A number of functions may be defined, these being:

(a) Cached DC delta - A delta is performed on the DC components using the stored values, and the delta values are then placed in the macroblock.

(b) DC quantise - a quantisation is performed only on the DC values in

20
the macroblock.

For each macro block that is not marked as dirty, but preceded by a macroblock marked as dirty since the start of the current slice, the following steps are performed:

(i)   restore the cached DC values;

(ii)   calculate the DC delta;

25
(iii) quantise the DC portion of the macroblock;

(iv)   encode the macroblock (bit stream encoding); and

(v)   save the bitstream encoded version.

An alternative method for caching and restoring macroblocks that are after a dirty block, but before the start of a new slice, involves the saving of the bitstream encoded version of the AC coefficients that comprise a macroblock separately from the DC coefficients. This is seen in Fig. 9C.

For blocks that are after the start of the slice (or are the start of a slice) and have not been preceded by a dirty block:

(i)   restore the DC portion of the bitstream; and

(ii)   restore the AC portion of the bitstream;

or, if the bitstream is cached as DC+AC, DC, AC:

(i)   restore the DC+ AC cached bitstream.

For blocks that are dirty:

(i)   apply DCT;

(ii)   calculate ΔDC and save the DC values;

(iii)   quantise;

(iv)   encode and save the DC values; and

(v)   encode and save the AC values.

For blocks that are not dirty but are preceded by a dirty block in the same slice:

(i)   calculate ΔDC;

(ii)   quantise the ΔDC;

(iii)   encode and save the DC; and

(iv)   restore AC bitsream.

Short-Circuit of DCT

Given that each macroblock that has been updated can be marked with the type of update that has been performed, it is possible to thereby identify the set of macroblocks that have been updated by way of a plane-fill. It is possible to efficiently transform these macroblocks by using a simplified version of the DCT, since it is known in advance that

5   the resulting transform domain representation will consist of a single value (namely the DC level for each macroblock).

Macroblock Level Screen Transitions

Related to the ability to save and restore I-Frame representations of main screens required by applications 210, is the ability to construct transitions between cached I-

10   Frame representations based on their macroblock representations.

This may be achieved by restoring a cached I-Frame into a temporary transition buffer, for example within the encoder 226, and then performing macroblock-sized operations (with associated encoding between a specified number of operations) to cause the contents of the transition buffer to be copied into a frame buffer, being another part of

15   the encoder 226. This can be achieved with high efficiency, given that both representations of an I-Frame contain cached macroblocks in both bitstream and quantised forms that may be encoded into a complete I-Frame with little overhead.

Some of examples of transitions that may be possible using this technique are:

(a)   mosaic effect wherein a certain number of macroblocks are pseudo-

20   randomly copied from the transition buffer to the frame buffer between the generation of each frame;

(b)   slide-on/slide-off effect wherein macroblocks in the target frame are migrated towards one edge of the buffer, with macroblocks from the transition frame copied to fill the space created; and

25   (c)   a spiral effect wherein macroblocks from the transition frame are copied in a spiral pattern into the target frame.

Motion Prediction From Affine Transform Matrices

Given that the hierarchical render-tree 250 is made up of nodes, each of which may be associated with a number of transformation matrices, it is possible to exploit the information provided by a composition of the transformation matrices to provide estimates for motion vectors (or in some cases serve directly as motion vectors) at macroblock level granularity. These motion vectors would form the principal data needed to encode full motion-predicted P-Frames.

On average, computing motion vectors directly for at 360x240 pixel resolution involves approximately 10^{10} operations. Even with fast search algorithms, this is only reduced to 10^9 or 10^8 operations. By making use of the hints provided by affine transform matrices, the number of operations may be reduced, in some cases, substantially.

Insertion of a Pre-Encoded MPEG Stream

For some applications, it may be desirable to display a video stream that is comprised of graphics not generated from the tree 250 or package 224, but for example directly from the process 222, as part of the output. Such may be accommodated by incorporation of the ability to insert, on-the-fly, one or more separate pre-encoded MPEG streams into a stream generated by the encoder 226.

To achieve this, firstly an area consisting of an integral number of macroblocks of a frame output from the encoder 226 is set aside for video insertion (the macroblocks that comprise this area could be specially flagged so that they would be treated specially during an encoding pass). The MPEG stream to be inserted must be partially decoded to the extent that the macroblocks representing each frame are extracted, and manipulated so that they are in a format suitable for I-Frame re-encoding. In some instances it may also be necessary to discard a certain number of macroblocks, since the insertion area, and the MPEG stream to be inserted, may not span equal numbers of macroblocks.

Once this has been done, inserting the semi-decoded video stream may be performed by simply copying macroblocks to their corresponding locations within the output frame for each frame update (and adjusting the DC delta coding accordingly), and then encoding an I-Frame as a whole from the output frame as described above, the

5   output frame being that provided from the application 210 to the set-top box 113.

Object Referenced Variable Quantisation

The various MPEG standards are flexible with regards to quantisation in that they allow variation of the quantisation matrix for each slice that comprises a frame. Since it is possible to obtain details of the exact buffer location of a set of composited

10   objects using the buffer changes output 244 of the graphics package 224, this information may be used to dynamically adjust the amount of quantisation applied to slices that correspond to certain composited objects.

For instance, if it was known that a particular region of the screen was comprised of composited objects that did not have significant edges, or other sharp

15   features/boundaries, then a large amount of quantisation could be applied (with a corresponding compression gain). Conversely, if a particular screen region was comprised of composited objects that had significant edges, or other sharp features/boundaries (such as text), then a small amount of quantisation could be applied, resulting in superior decoded picture quality.

20   Encoding System

A specific advantage in the use of the MPEG encoding system is that in the various MPEG standards, the standard is defined at the stream level rather than at the encoder level. As a consequence, system manufacturers and program material developers are free to develop MPEG applications that can operate in different ways provided that,

25   when delivered to an appropriate MPEG decoder, that decoder is able to properly decode the stream. As a consequence, the standards of the MPEG system can be implemented

within simple set-top boxes to provide for traditional MPEG decoding. Further, such permits the set-top box 114 to be modified and enhanced using application specific hardware and software to achieve modified forms of decoding essentially consistent with but providing alternate performance beyond that of traditional MPEG systems.

5   It follows therefore that the encoder 226 may be developed so as to apply any one or more of the foregoing methods of optimisation to achieve enhanced MPEG encoding of information required to be delivered to the user within the system 100 and, in particular, in response to the user's input commands. Such an arrangement finds particular utility in relation to those outputs that are essentially static, such as the menu

10   driven displays of Figs. 1B and 1C, or where the rate of update is small compared to continuously streamed video (whether encoded or not), such as shown in Fig. 1D. Such an arrangement is characterised in that it is the rendering from the graphics package that creates the encoded MPEG stream (ie. the rendering is encoded as a stream). The methods forming the MPEG stream described herein result in the generation of a stream

15   in response to user's commands and that is consistent with the MPEG standards and thus able to be decoded using generic equipment formed within the set-top box 114. This to be contrasted with prior art arrangements where extra data is packed into an existing MPEG stream (ie. rendering into the stream). Such prior art arrangements typically require proprietary hardware and/or software to filter out or otherwise extract the extra

20   data at the decode end of the system (ie. the set-to box).

         A specific application of the above described encoding arrangements and which may be implemented using the configurations of Figs. 1 and 2, relates to browsing of the network 105 or of another network, where the browser is an application 210 operating from within the server computer 118. In this regard, where the network 105 is a special

25   purpose network such as a cable-TV distribution arrangement, the "other" network may be the Internet, for example separately connected to the server 118. Alternatively, the

network 105 may be the Internet or some similar arrangement. An advantage of such an arrangement is that it facilitates web-browsing without requiring the user to invest in substantial hardware and software products. By removing the web-browser and image generation functions to the server 118, enable the user-end of the system, comprising the

5   set-top box 114, remote controller 106 and output device 116, to be formed of relatively inexpensive devices capable of simple operation and therefore offering substantially improved reliability and reduce start-up costs. The arrangement also permits the operator of the server computer 118, such as an Internet Service Provider or cable TV operator, to upgrade and/or modify the capabilities and power of the computer 118 without requiring

10   alteration or upgrade of the user-end of the system.

The methods of encoding described above are preferably practiced using a general-purpose computer system 1300, such as that shown in Fig. 13 wherein the processes of Figs. 2 to 12 may be implemented as software, such as one or more application program executing within the computer system 1300. In particular, the steps

15   encoding are effected by instructions in the software that are carried out by the computer. The software may be stored in a computer readable medium, including the storage devices described below, for example. The software may be loaded into the computer from the computer readable medium, and then executed by the computer. A computer readable medium having such software or computer program recorded on it is a computer

20   program product. The use of the computer program product in the computer preferably effects an advantageous apparatus for encoding and decoding.

The computer system 1300 comprises a computer module 1301, input devices such as a keyboard 1302 and mouse 1303, output devices including a printer 1315 and a display device 1314. A Modulator-Demodulator (Modem) transceiver device 1316 is

25   used by the computer module 1301 for communicating to and from a communications network 1320, for example connectable via a telephone line 1321 or other functional

medium. The modem 1316 can be used to obtain access to the Internet, and other network systems, such as a Local Area Network (LAN) or a Wide Area Network (WAN).

The computer module 1301 typically includes at least one processor unit 1305, a memory unit 1306, for example formed from semiconductor random access memory (RAM) and read only memory (ROM), input/output (I/O) interfaces including a video interface 1307, and an I/O interface 1313 for the keyboard 1302 and mouse 1303 and optionally a joystick (not illustrated), and an interface 1308 for the modem 1316. A storage device 1309 is provided and typically includes a hard disk drive 1310 and a floppy disk drive 1311. A magnetic tape drive (not illustrated) may also be used. A CD-ROM drive 1312 is typically provided as a non-volatile source of data. The components 1305 to 1313 of the computer module 1301, typically communicate via an interconnected bus 1304 and in a manner which results in a conventional mode of operation of the computer system 1300 known to those in the relevant art. Examples of computers on which the described arrangements can be practised include IBM-PC's and compatibles, Sun Sparcstations or alike computer systems evolved therefrom.

Typically, the application program is resident on the hard disk drive 1310 and read and controlled in its execution by the processor 1305. Intermediate storage of the program and any data fetched from the network 1320 may be accomplished using the semiconductor memory 1306, possibly in concert with the hard disk drive 1310. In some instances, the application program may be supplied to the user encoded on a CD-ROM or floppy disk and read via the corresponding drive 1312 or 1311, or alternatively may be read by the user from the network 1320 via the modem device 1316. Still further, the software can also be loaded into the computer system 1300 from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer module 1301 and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets

including e-mail transmissions and information recorded on Websites and the like. The foregoing is merely exemplary of relevant computer readable media. Other computer readable media may alternately be used.

The various applications 210 as discussed above may operate within a server

5   computer 118 which may include many of the traditional features found in computer like devices such as the computer system 1300 shown in Fig. 13. Further, the set-top box 114 may include much of the arrangement of the computer module 1301 of Fig. 13 noting that in such arrangements, typically a floppy disk drive, hard disk drive, or CD ROM drive would not typically be required. In particular, where the computer system 1300 of Fig. 13

10 is configured to operate the applications 210, various encoding steps may be performed within the computer module 1301. Similarly, where such arrangement is formed within the set-top box 114, corresponding decoding arrangements may be performed thereby.

The encoding and decoding methods described may alternatively be implemented in dedicated hardware such as one or more integrated circuits performing

15 the functions or sub functions described above. Such dedicated hardware may include graphic processors, digital signal processors, or one or more microprocessors and associated memories and operate in concert with software components.

Industrial Applicability

It is apparent from the above that the arrangements described are applicable to

20 the computer and data processing industries and particularly where data from a variety of data sources is being encoded or compressed.

The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiments being illustrative and not restrictive.

25 AUSTRALIA ONLY   In the context of this specification, the word "comprising" means "including principally but not necessarily solely" or "having" or

"including" and not "consisting only of". Variations of the word comprising, such as

"comprise" and "comprises" have corresponding meanings.

The claims defining the invention are as follows:

1.   A method of encoding a sequence of images for transmission over a communication network, said method comprising the steps of:

5   (i)   rendering a first image from a computer graphics application;

          (ii)   encoding said first image according to a predetermined encoding scheme;

          (iii)   rendering a second image from said computer graphics application;

          (iv)   identifying changes between said first image and said second image

10   from a change indicating output of said computer graphics application;

          (v)   using said identified changes to determine a manner in which said second image is encoded according to said predetermined format; and

          (vi)   repeating steps (iii) to (v) for each subsequent image of said sequence.

15   2.   A method according to claim 1 wherein said computer graphics application outputs a pixel-based representation of each said image to a buffer, said encoding operating, at least for said first image, upon said pixel-based representation and for said second and subsequent images upon at least said change indicating output.

20   3.   A method according to claim 2 wherein said change indicating output comprises information indicating an extent of change in said pixel-based representation in said buffer thereby enabling said encoding according to said predetermined encoding scheme of substantially only those pixels that have changed.

25   4.   A method according to claim 2 or 3 wherein said computer graphics application acts to render a hierarchical representation of each said image and said change indicating

output comprises information indicating an extent of change between said images of the corresponding said hierarchical representations.

5.   A method according to any one of the preceding claims wherein said predetermined encoding scheme comprises MPEG encoding and said encoding forms, for each said image, a data stream for transmission over said network.

6.   An interactive system in which user commands at a first location are communicated via a communications network to a processing arrangement at a second location, said processing arrangement interpreting said commands and in response thereto transmitting a sequence of images for display at said first location to said user, each said image being encoded to form at least a component of a data stream, characterised in that each said image is formed at said processing arrangement from a graphic object representation of said image and a rendering of said graphic object representation is used to encode said image and form said stream.

7.   A system according to claim 6 wherein said rendering is used exclusively as an encoding source and from which said stream is formed.

8.   An encoder for encoding a series of images into a bitstream, each said image being rendered from a graphics object application characterised in that said encoder is constrained to operate according to a plurality of inputs at least one of which comprising a bitmap of each said image.

9.   An encoder according to claim 8 wherein one of said inputs comprises a first change input representing those portions of said bitmap of a current said image have changed relative to an immediately preceding image in said series.

5   10.   An encoder according to claim 8 or 9 wherein an input to said graphical object application comprises a hierarchical representation of said image and one of said inputs comprises a second change input representing those portions of said hierarchical representation of a current said image have changed relative to an immediately preceding image in said series.

10

11.   A system for manipulating user commands over a computer network, said system comprising:

        a user terminal coupled to said network at a first location and incorporating means for receiving commands from said user and display means for displaying images;

15   a server computer coupled to said network at a second location, said server computer comprising a computing application arranged to receive and process said commands from said user terminal, and an output generator for forming an output display of said computing application;

        characterised in that

20   said output generator comprises a graphic object application for rendering images formed by said computing application and an encoder for forming an encoded bitstream output of said rendered images for communication over said network for display on said display means of said user terminal, and said user terminal comprises a decoder for decoding said bitstream for display on said display means.

25

12.   A system according to claim 11 characterised in that said computing application comprises a browsing application for browsing said computer network.

13.   A system according to claim 11 or 12 characterised in that said bitstream

5   comprises an MPEG stream.

14.   A system according to claim 13 characterised in that said bitstream consists of an MPEG stream.

10   15.   The invention according to any one of claims 5, 7, 10 or 14 wherein said encoding comprises MPEG encoding and incorporates at least one of the optimisations substantially as described herein.

16.   A method of encoding a sequence of images for transmission over a

15   communication network, said method being substantially as described herein with reference to Fig. 2 and at least one of the optimisations described with reference to Figs. 3 to 12 of the drawings.

17.   An interactive system in which user commands at a first location are

20   communicated via a communications network to a processing arrangement at a second location, said processing arrangement interpreting said commands and in response thereto transmitting a sequence of images for display at said first location to said user, said system being substantially as described with reference to the drawings.

25   18.   An encoder for encoding a series of images into a bitstream substantially as described herein with reference to Fig. 2 and at least one of Figs. 3 to 12 of the drawings.

19.   A bitstream produced using the invention of any one of the preceding claims.

20.   A web-browsing system substantially as described herein with reference to

5   Figs. 2 to 12 of the drawings.

Dated this EIGHTEENTH Day of DECEMBER 2000

CANON KABUSHIKI KAISHA

Patent Attorneys for the Applicant

10   SPRUSON&FERGUSON

118

105

100

Network

STB

114

Audio/Visual Output

116

108

110

102

MusicOnLine Blues Guitar Masters

104

112

106

Fig. 1A (Prior Art)

120

MusicOnLine
Blues Guitar Masters
Please make a selection
Old Dead Guy ☐
Old Living Man ☐
Living Legend ☐
Young Dead Guy ▨

Fig. 1B

122

MusicOnLine
Blues Guitar Masters
Young Dead Guy
Live at Carnegie Hall ☐
Live in the TV Studio ▨
Music Videos ☐
Interview ☐

Fig. 1C

124

Fig. 1D

514988.fm
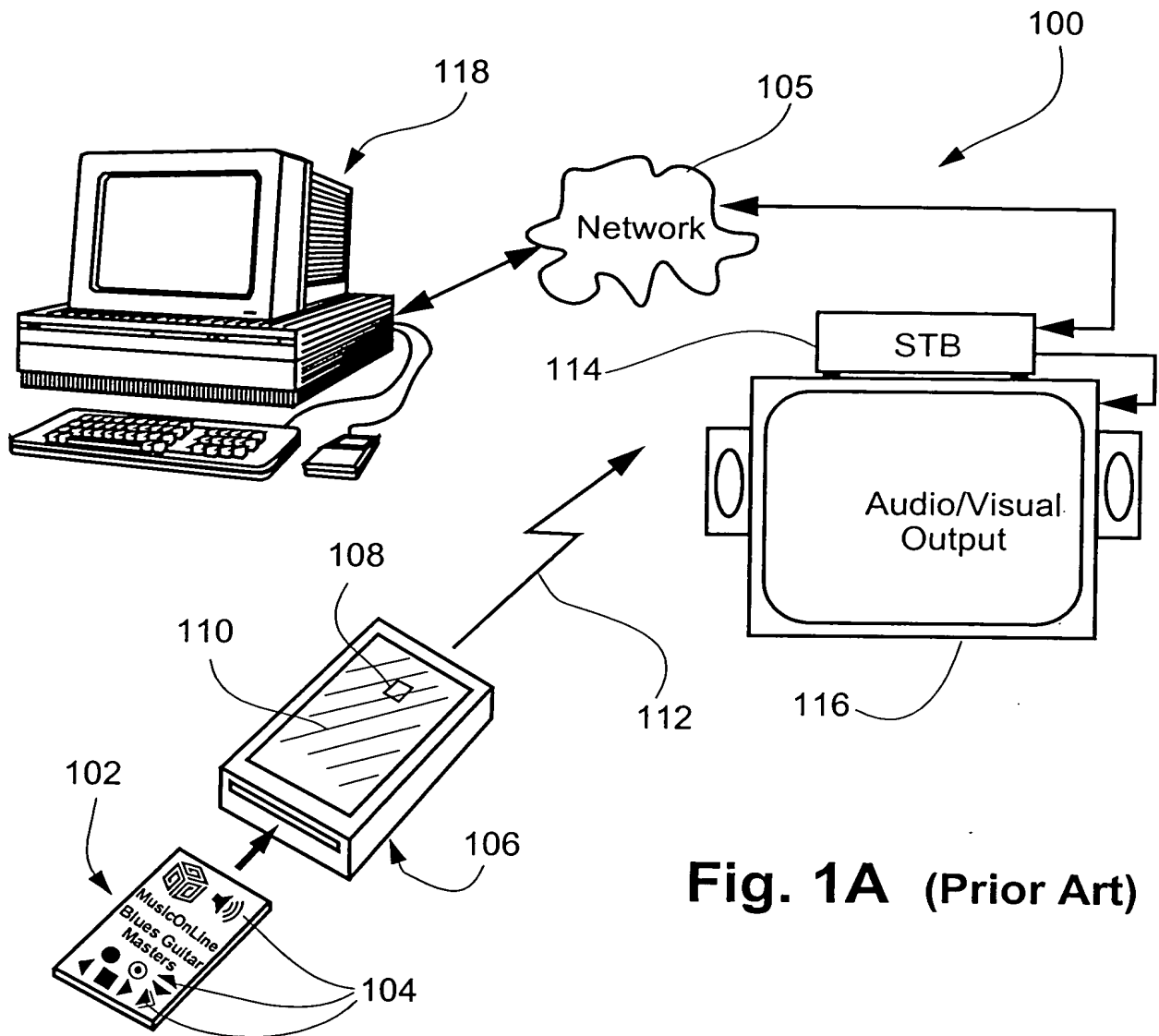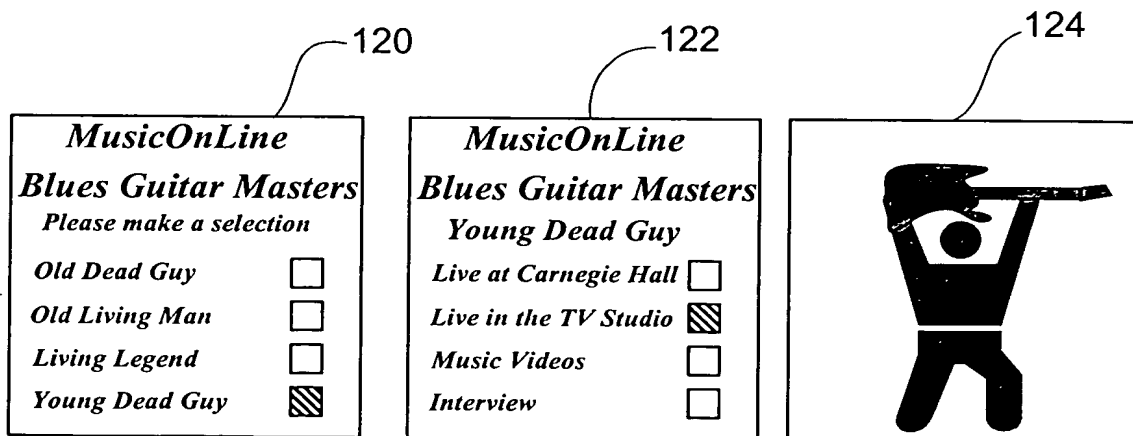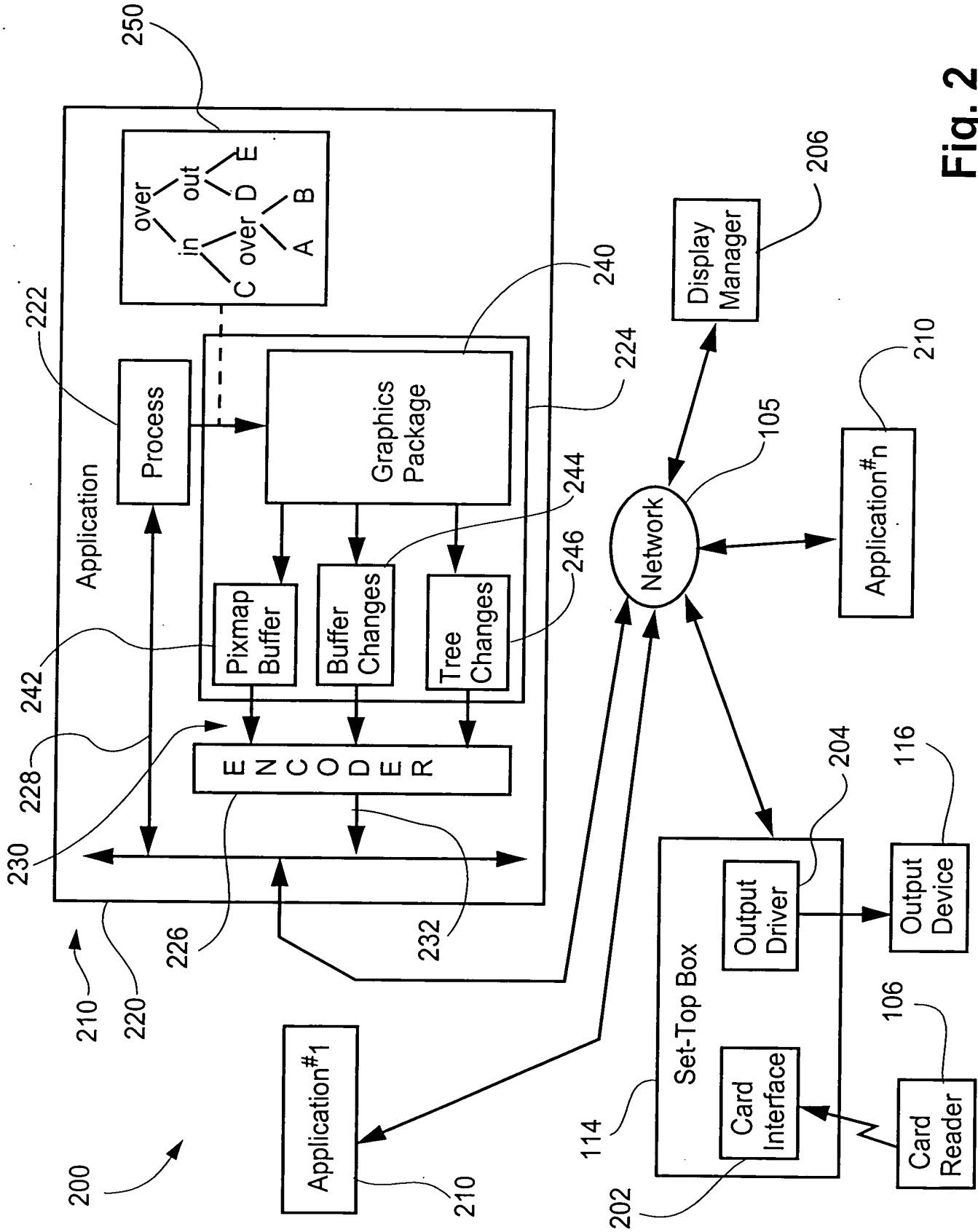
Fig. 2
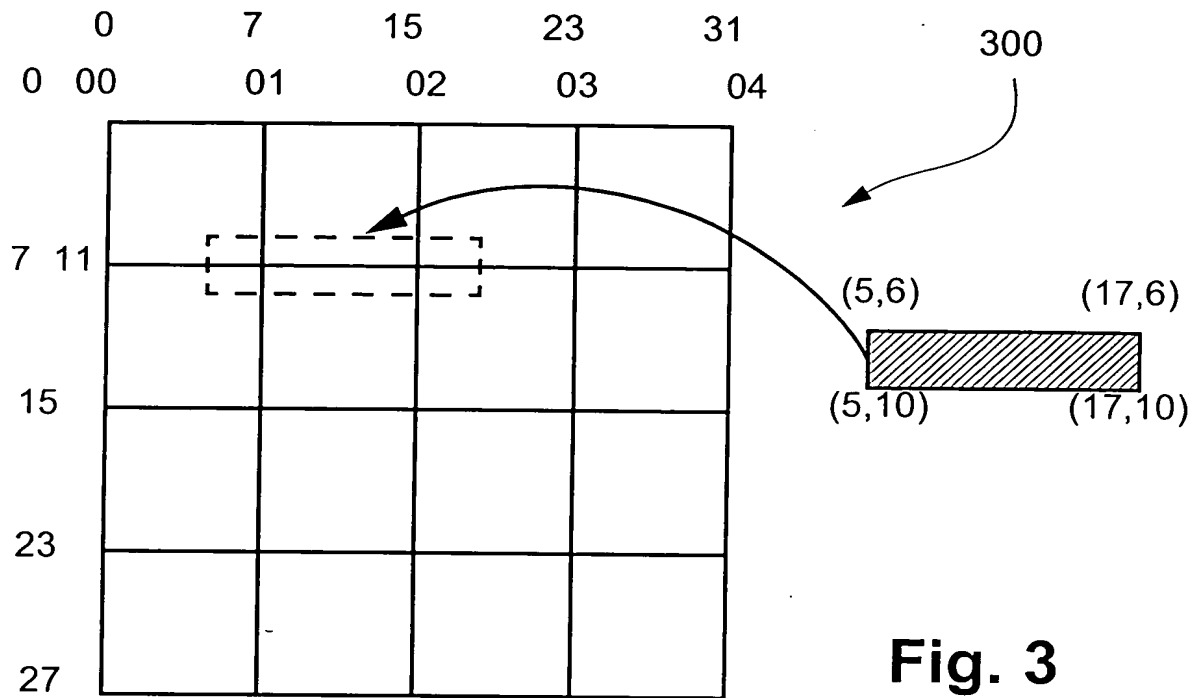
Fig. 3
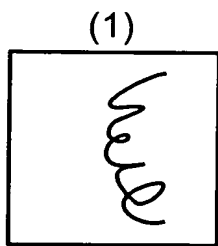
Fig. 4A   Fig. 4B   Fig. 4C   Fig. 4D

Fig. 4E

500

502

Fig. 5

Bounds
Dirty Flag
Encoded block
Encoded bit offset
Encoded bit count

602

600

604
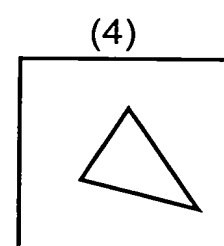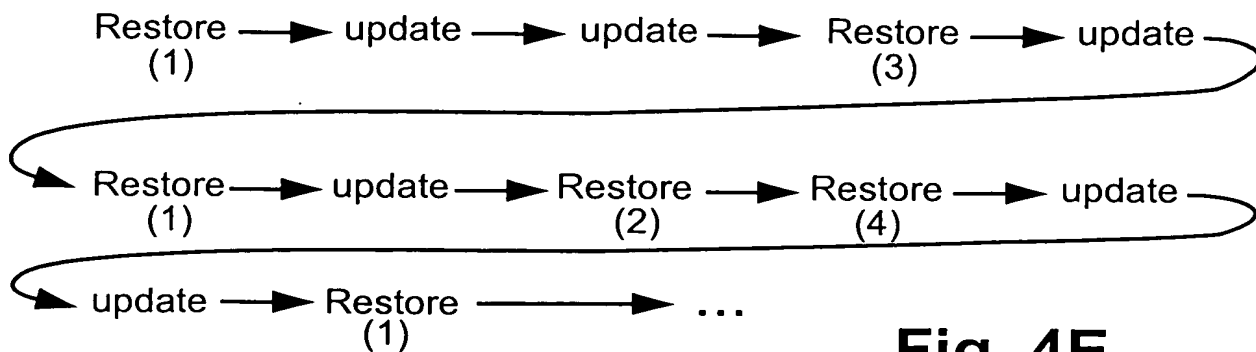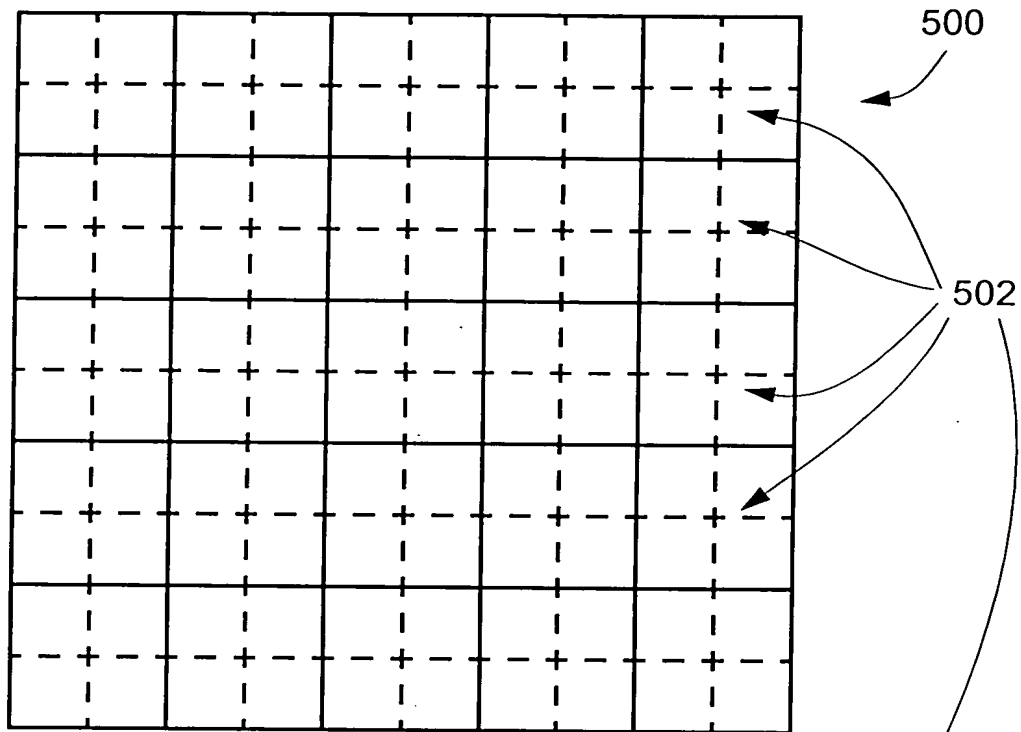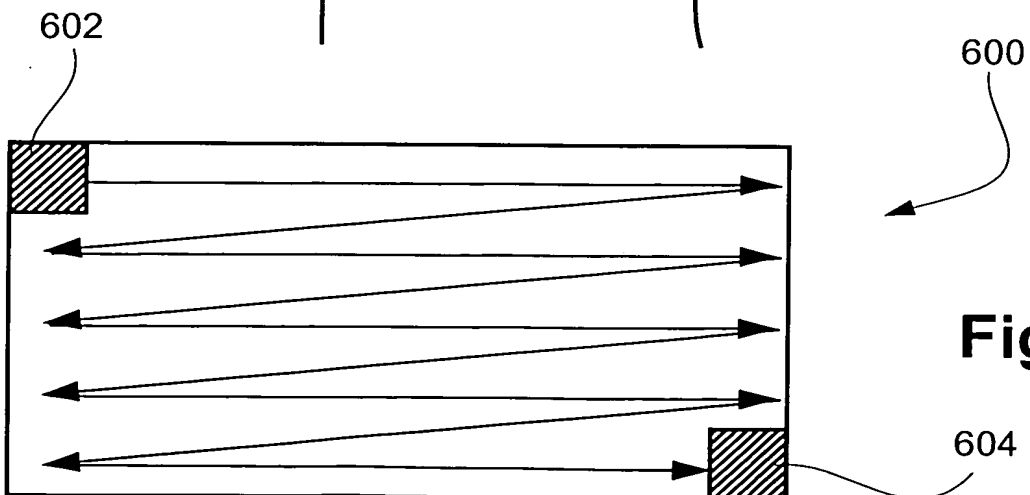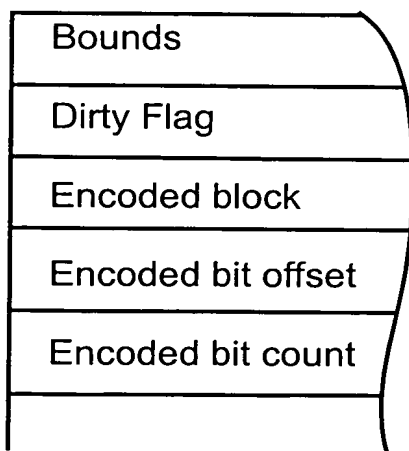
Fig. 6
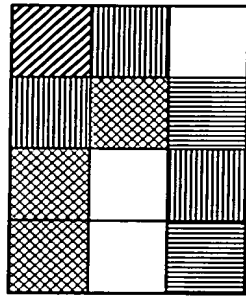
RGB ——→ YCbCr for each pixel then DCT

Fig. 7A
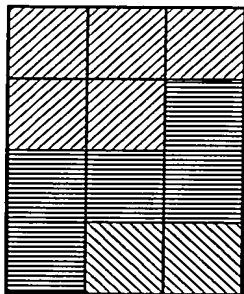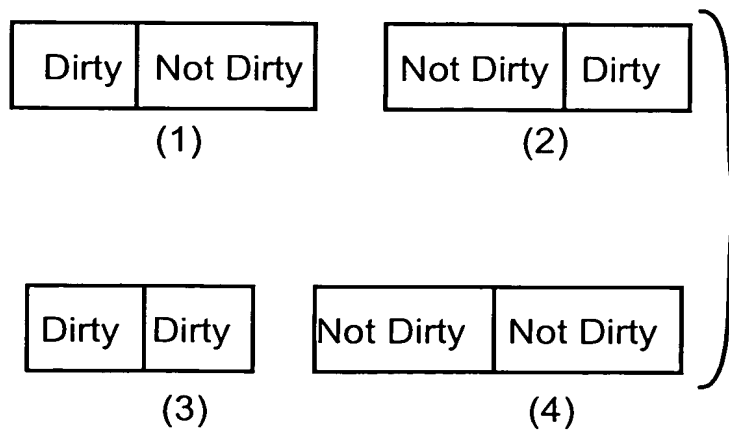

RGB ——→ YCbCr for each run then DCT

Fig. 7B
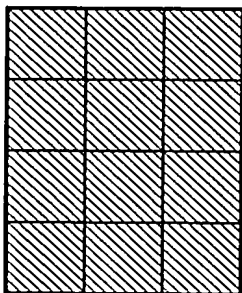
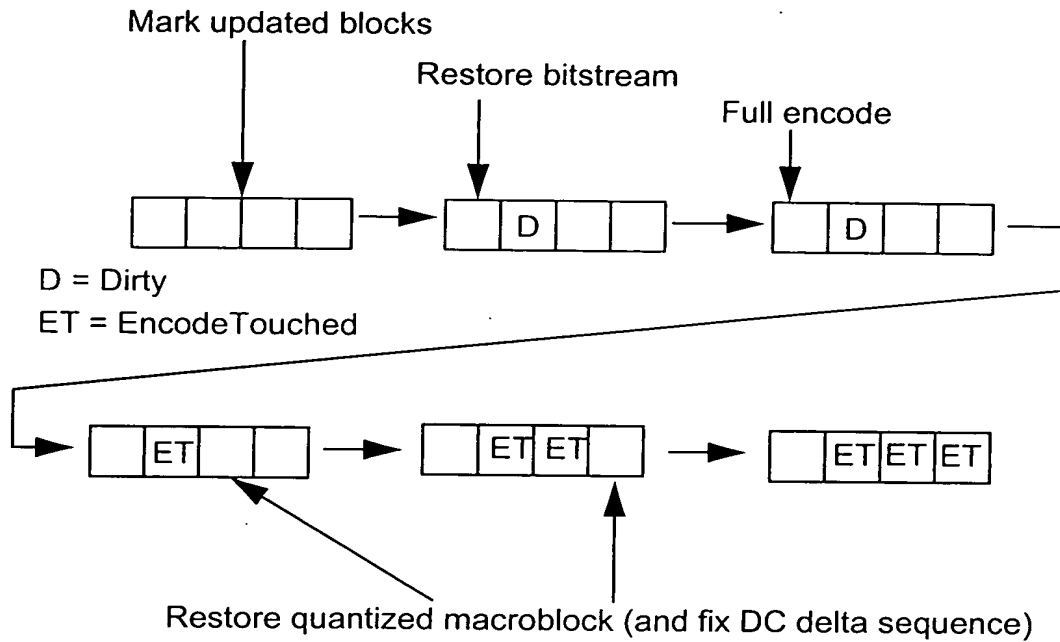RGB ——→ YCbCr for a pixel
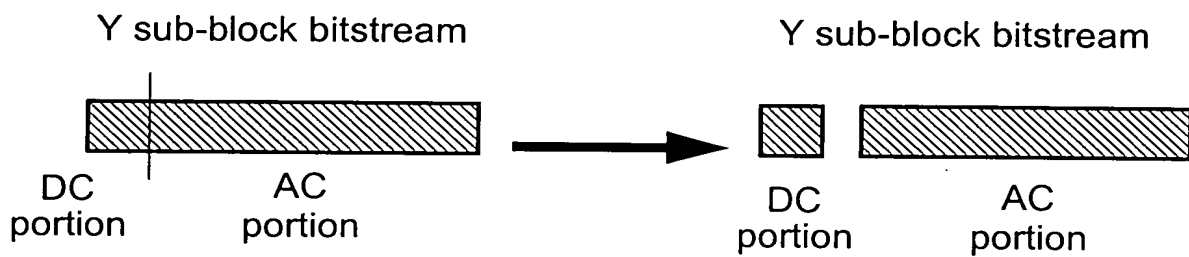
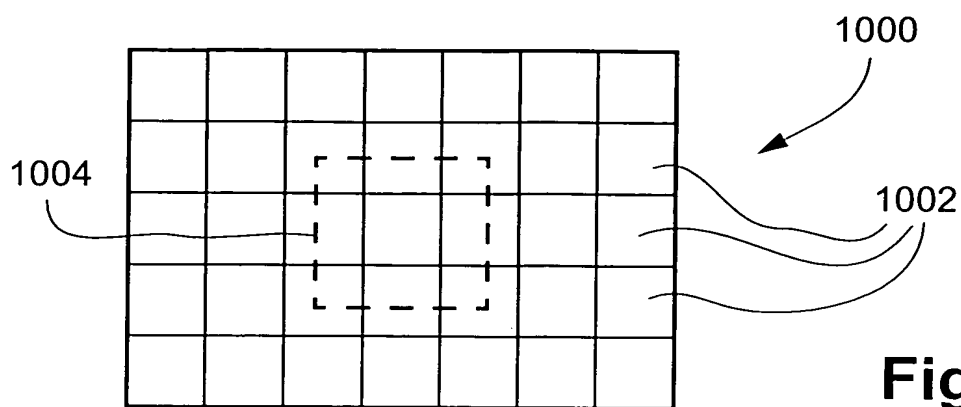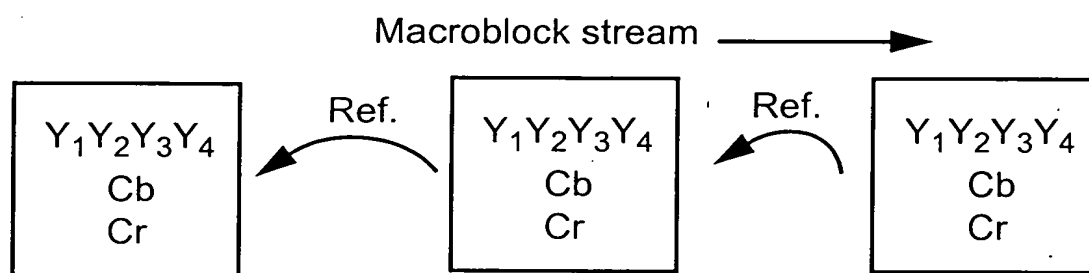# Fig. 7C

Fig. 8

Fig. 9A

Mark updated blocks

Restore bitstream

Full encode

D = Dirty
ET = EncodeTouched

Restore quantized macroblock (and fix DC delta sequence)

Fig. 9B

Y sub-block bitstream

Y sub-block bitstream

DC portion   AC portion

DC portion   AC portion

Fig. 9C

514988.fm

Fig. 10

Macroblock stream →

Fig. 11

Fig. 12

Fig. 13